

# La mémoire partagée distribuée

- [Pourquoi une mémoire partagée distribuée ?](#)
  - [Tightly coupled shared-memory multiprocessors](#)
  - [Distributed-memory multiprocessors](#)
  - [Avantages / inconvénients de la mémoire distribuée](#)
  - [Implémentation d'une mémoire distribuée](#)
  - [Granularité des pages](#)
  - [Partage des objets](#)
  - [Solution centralisée](#)
  - [Autres solutions](#)
  - [Site propriétaire](#)
- 

## Pourquoi une mémoire partagée distribuée ?

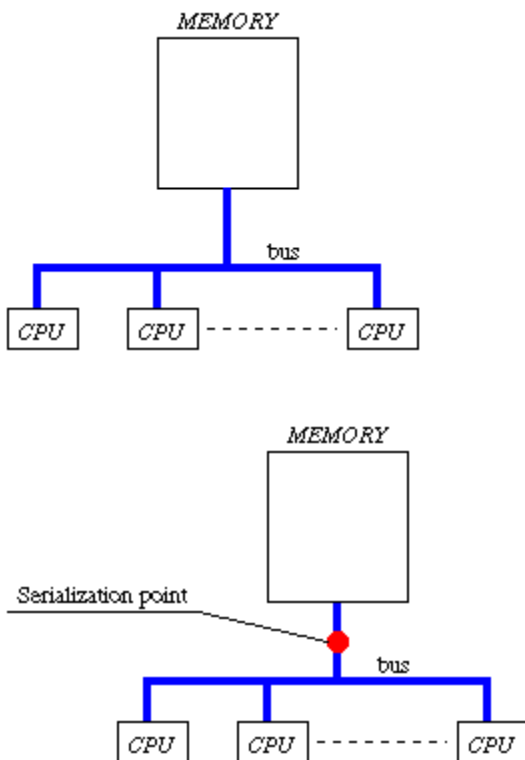
Limite **physique** des processeurs et mémoires.

⇒ utilisation de multi-processeurs pour accroître la puissance de calcul.

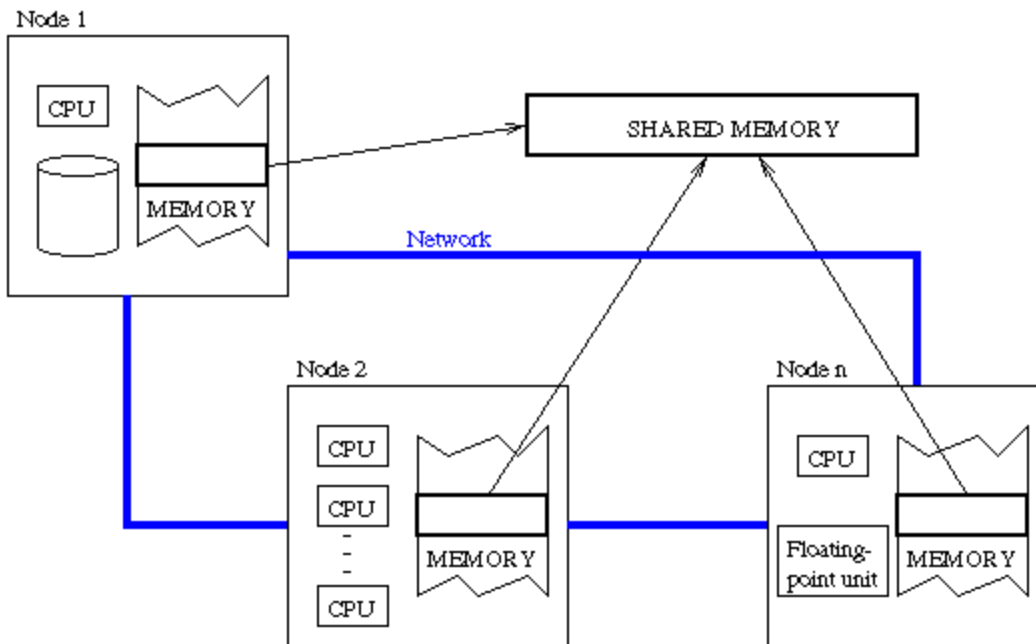
2 sortes de processeurs parallèles :

- **tightly coupled shared-memory multiprocessors;**
- **distributed-memory multiprocessors.**

## Tightly coupled shared-memory multiprocessors



## Distributed-memory multiprocessors



## Avantages / inconvénients de la mémoire distribuée

Avantages :

- Illusion d'une mémoire physique partagée, sans bottleneck.
- Scalability (on peut étendre le système sans trop de difficulté).
- coût moindre.

Inconvénients :

- Topologie du réseau très importante.
- Gestion du réseau.

## Implémentation d'une mémoire distribuée

Accès mémoire partagé → Communication Inter-processus.

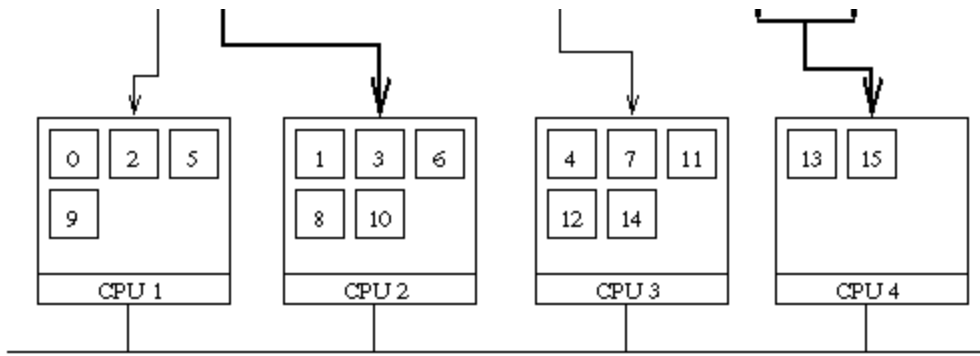
Aucun processeur ne peut directement accéder à la mémoire d'un autre processeur → NORMA (NO Remote Memory Access) systems.

Les processeurs référencent leur propre mémoire locale. Il faut rajouter du software pour que lorsqu'un processeur référence une page distante, cette page soit récupérée.

L'espace d'adressage commun est découpé en morceaux.

Chaque morceau est situé sur une station.

Quand un processeur référence une page non-locale → "trap" (page fault) → DSM va récupérer la page.



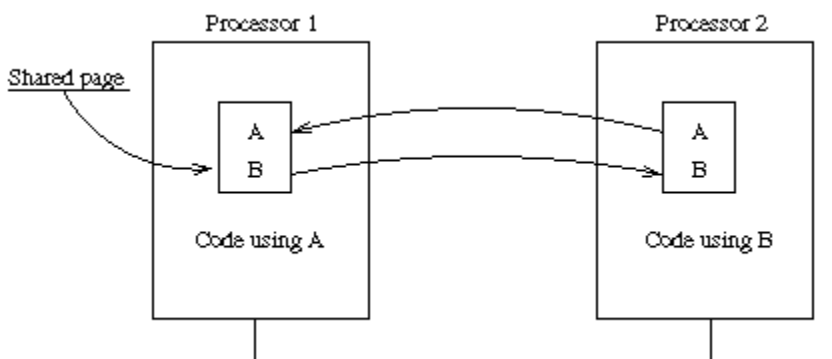
## Granularité des pages

Page de granularité importante, avantages :

- réduction du trafic sur le réseau ;
- propriété de localité.

Inconvénient :

Problème du **false sharing** :



## Partage des objets

Pour partager des objets, un processus doit pouvoir :

- trouver l'objet ;
- le récupérer.

Si l'objet reste toujours au même endroit (même station) ⇨ facile (répertoire).

Si l'objet peut **migrer** ⇨ plus compliqué.

## Solution centralisée

On a un serveur central qui garde trace de tous les déplacements d'objets.

⇨ on veut éviter ça.

- tolérance aux fautes minime (disponibilité minime) ;
- surcharge du serveur (performance minime) :
  1. réduction du parallélisme (sérialisation des requêtes),

2. ralentissement de tout le système.

## Autres solutions

Diffuser des messages de "Data Location".

Problème : la diffusion ne se prête pas à la scalability.

De plus, la **latence** du réseau peut engendrer des délais de transmission importants.

⇒ les systèmes utilisent un schéma de distribution basé sur le site propriétaire.

## Site propriétaire

1 objet = 1 site propriétaire (qui possède la copie primaire de l'objet).

Le propriétaire change quand l'objet se déplace.

Au départ, les propriétaires sont connus (broadcast). Quand un site a besoin d'un objet ⇒ requête au propriétaire. Si le propriétaire n'a plus l'objet, il fait suivre la requête.

Problème : si l'objet bouge souvent ⇒ beaucoup de "sauts".



[Retour au sommaire.](#)